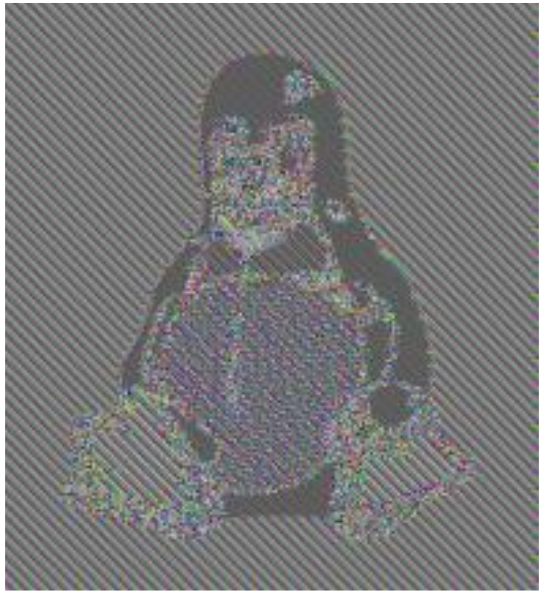
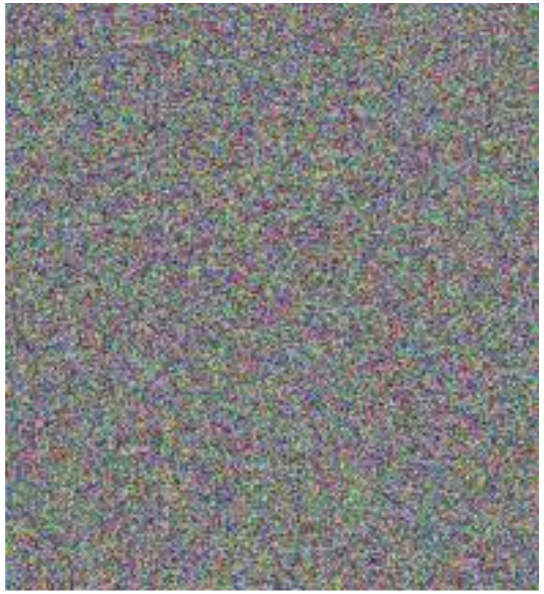
ECB weakness (Cond.)

Plaintext



ECB Mode Encryption



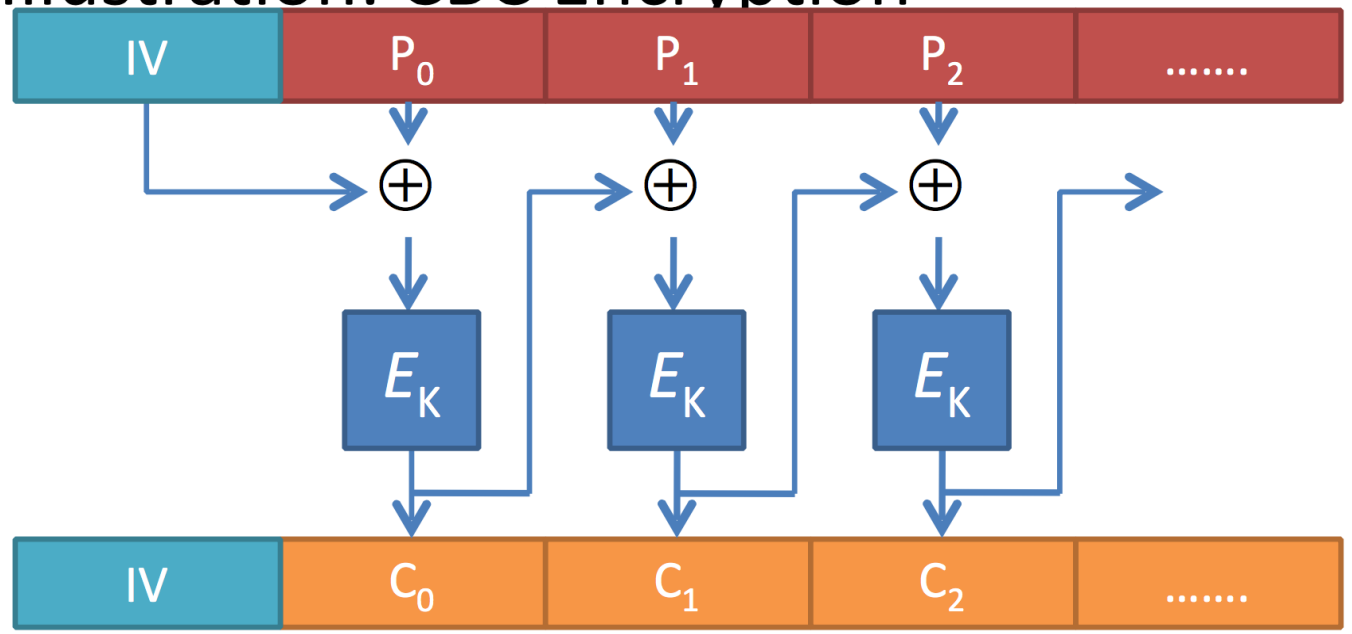
CBC/Other Modes

Chaining

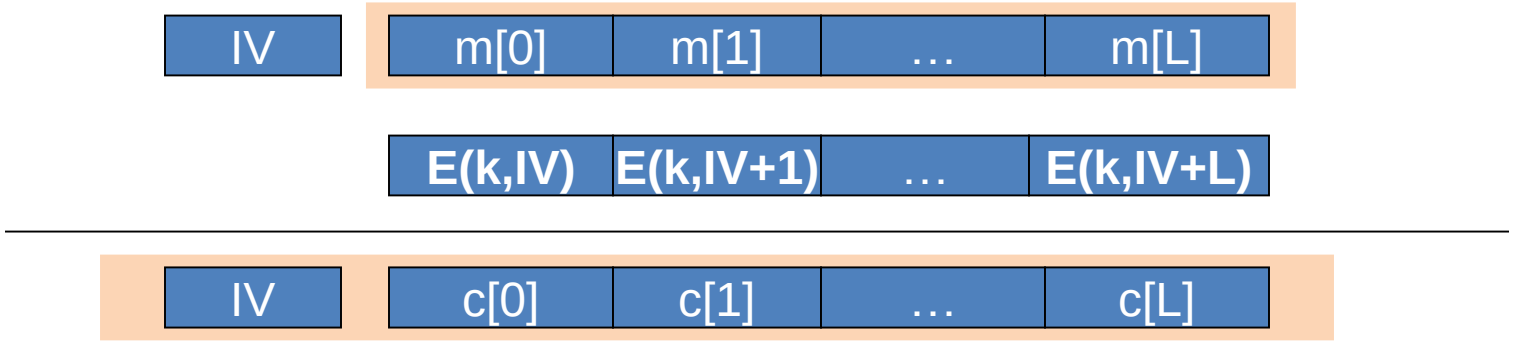
- Encryption of each block depends on previous blocks
- Problem: first block has no prior block?
  - Plaintext of first block can be inferred
- Solution: Initialization Vector (IV)
  - An extra block chained to the first block
  - Both parties must use the same IV
  - Must be unpredictable to adversary!

CBC (Cipher-block Chaining) mode

Illustration: CBC Encryption



CTR (Counter) mode



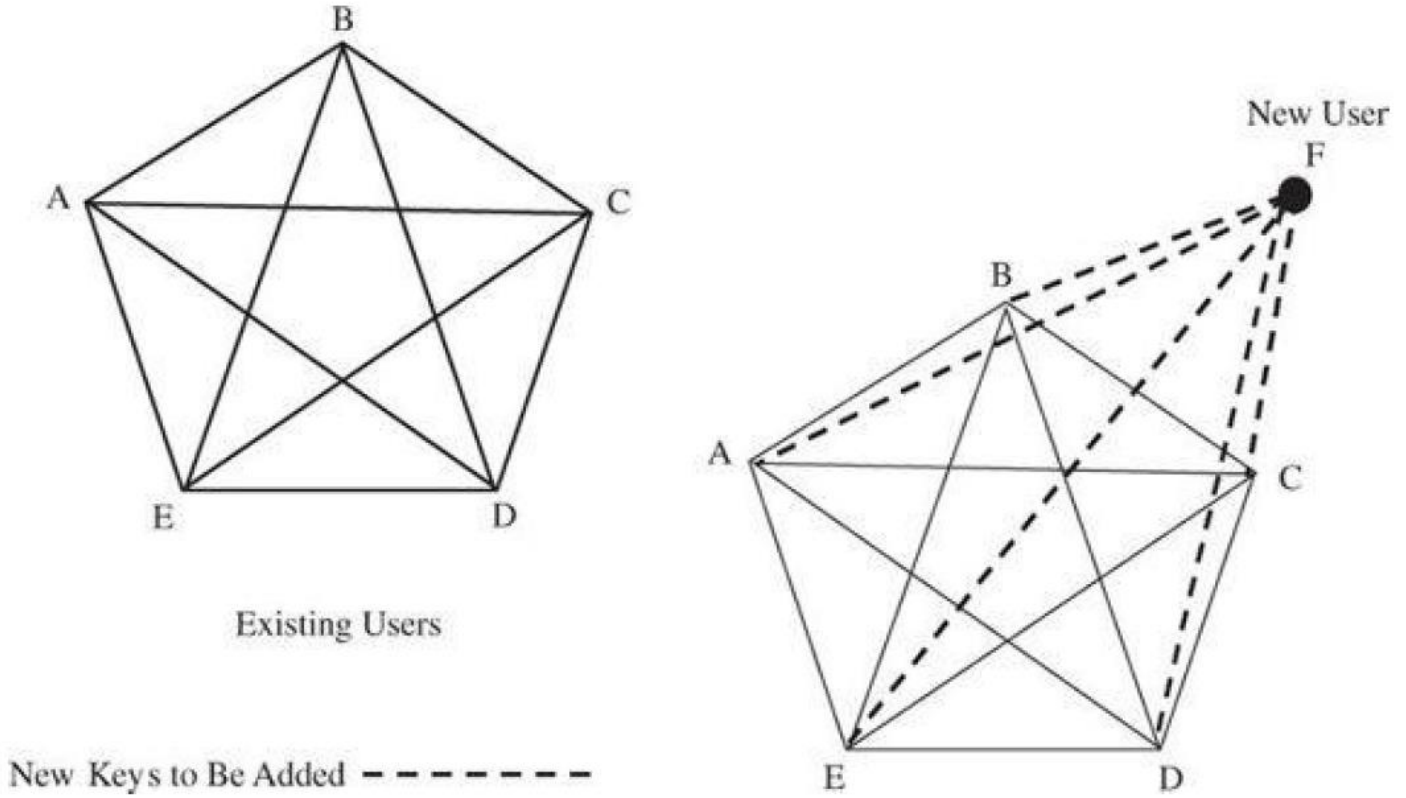
Block encryption & decryption can be parallelized

Stream vs. Block

	Stream	Block
Advantages	• Speed of transformation • Low error propagation	• High diffusion • Immunity to insertion of symbol
Disadvantages	• Low diffusion • Susceptibility to malicious insertions and modifications	• Slowness of encryption • Padding • Error propagation

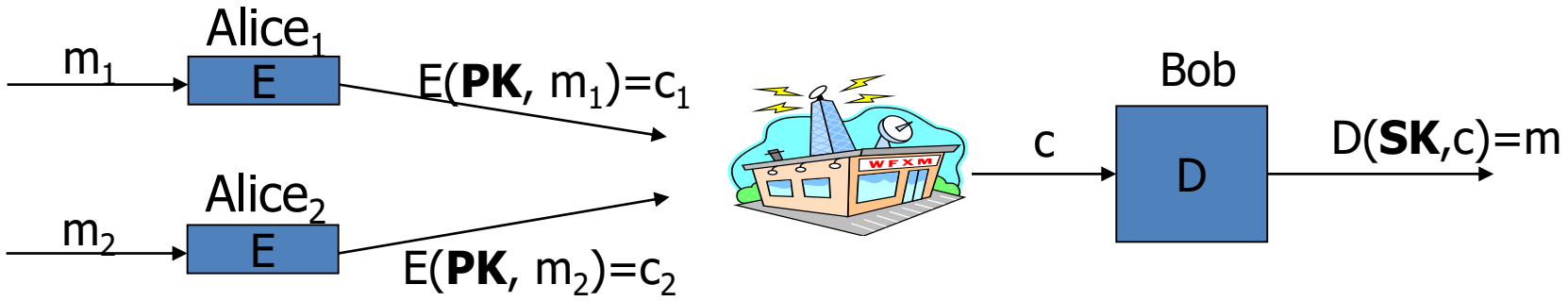
The key management problem

- *n*-user system requires *n* * (*n* – 1)/2 keys



Public Key (Asymmetric) Cryptography

- Proposed by Whitfield Diffie & Martin Hellman at 1976
- Instead of two users sharing one secret key, each user has two keys: one *public* and one *private*
- Messages encrypted using the user's public key can only be decrypted using the user's private key, and vice versa



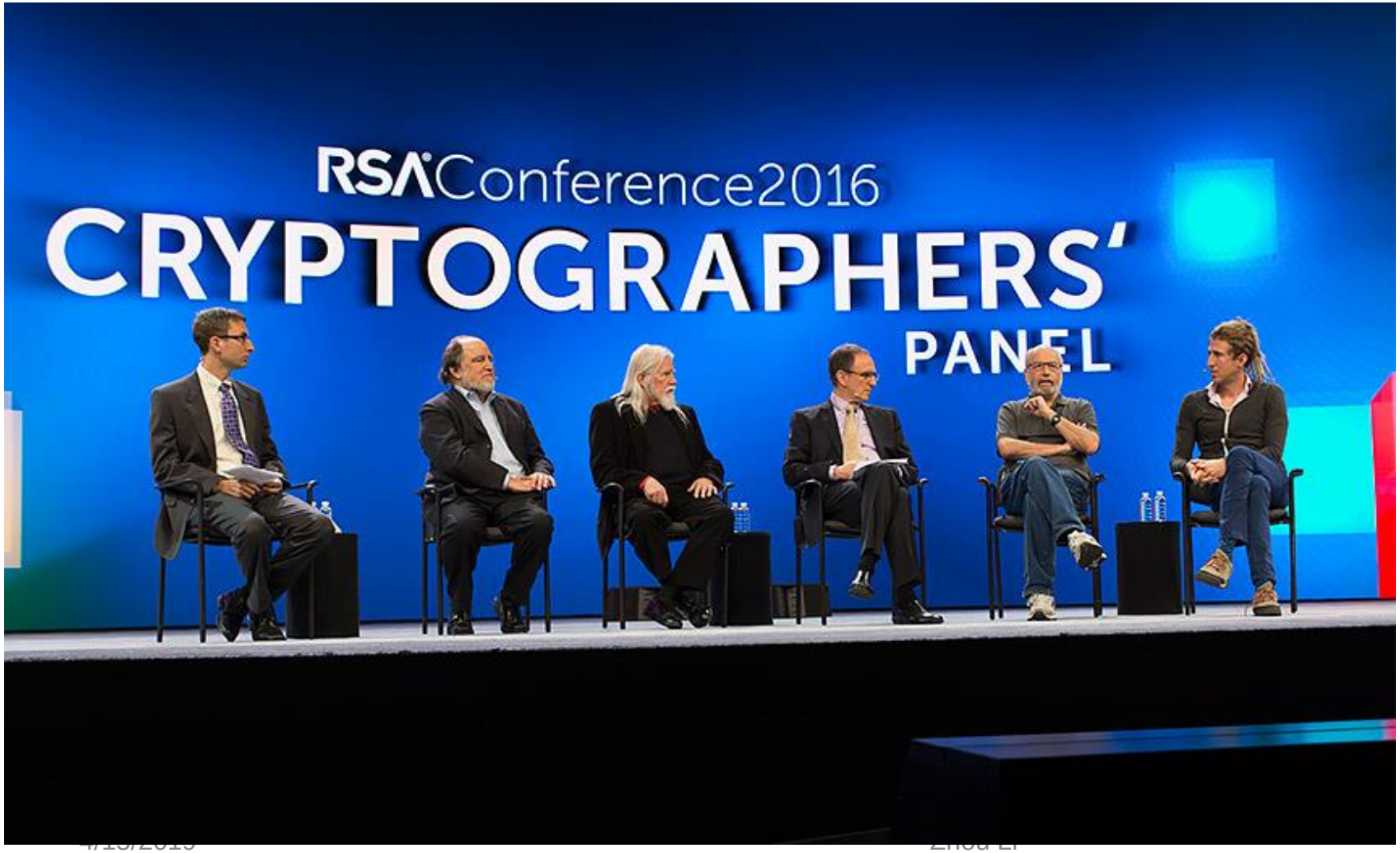
RSA

- Rivest-Shamir-Adelman cryptosystem
  - Ronald Rivest
  - Adi Shamir
  - Leonard Adleman
  - Introduced in 1978

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman*

They gather together nearly every year



Video of 2018:

https://www.rsaconference.com/videos/the-cryptographers-panel-2018

Asymmetric Encryption with RSA

- Since its introduction, RSA has been the subject of extensive cryptanalysis, and no serious flaws have yet been found
- The encryption algorithm is based on the underlying problem of factoring large prime numbers, a problem for which the fastest known algorithm is exponential in time
- Two keys, *(n,d)* and *(n,e)*, are used for decryption and encryption (they are interchangeable)

RSA scheme: Choosing keys

1. Choose two large prime numbers p, q.
   (e.g., 2048 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$ — totient function
3. Choose e (with $e<n$) that has no common factors with z.
   (e, z are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z.
   (in other words: $ed \bmod z = 1$).
5. *Public* key is (n,e). *Private* key is (n,d).

K_B^+ K_B^-

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

$c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

$m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

RSA: Why It works?

$$m = (m^e \bmod n)^d \bmod n$$

Useful number theory result: If p, q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

$c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

$m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

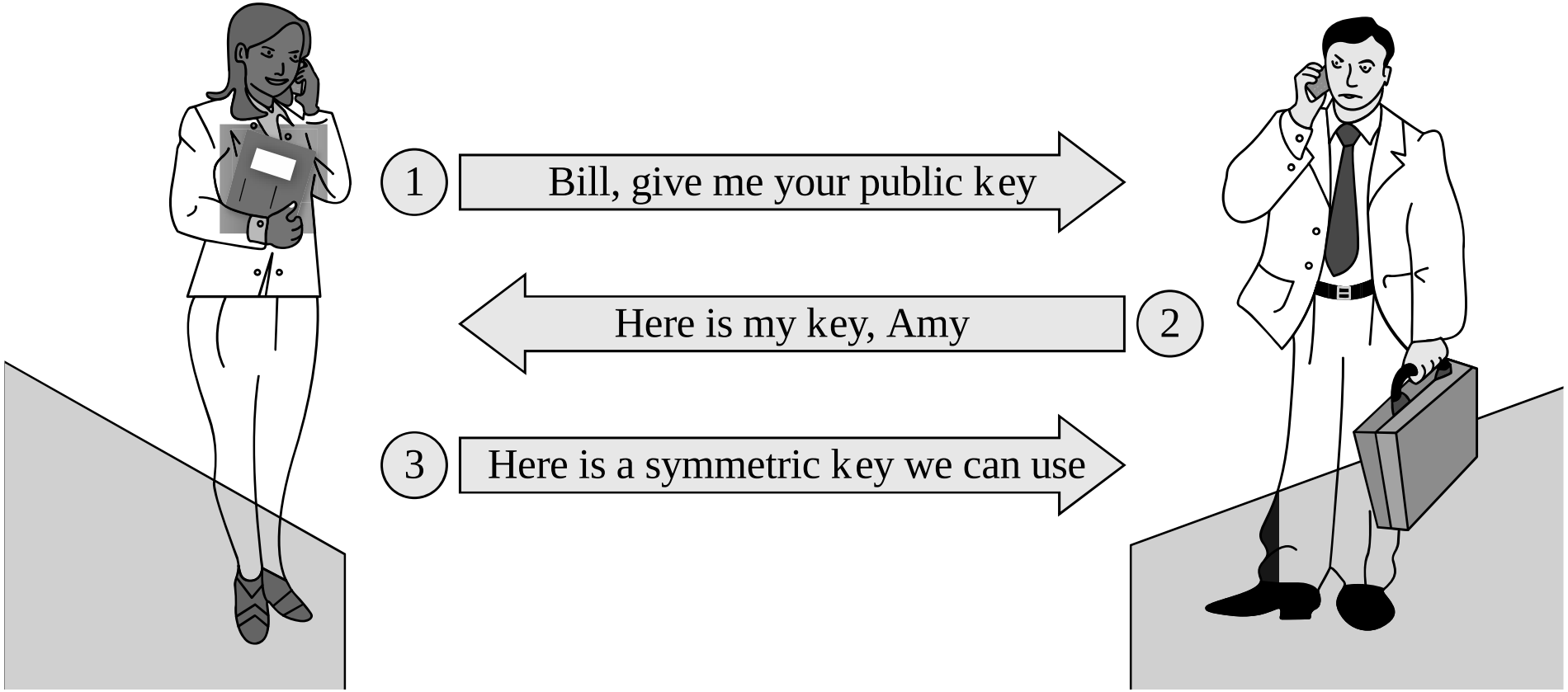
Magic happens!

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

Secret Key vs. Public Key Encryption

	Secret Key (Symmetric)	Public Key (Asymmetric)
Number of keys	1	2
Key size (bits)	56–112 (DES), 128–256 (AES)	Unlimited; typically no less than 256; 1000 to 2000 currently considered desirable for most uses
Protection of key	Must be kept secret	One key must be kept secret; the other can be freely exposed
Best uses	Cryptographic workhorse. Secrecy and integrity of data, from single characters to blocks of data, messages and files	Key exchange, authentication, signing
Key distribution	Must be out-of-band	Public key can be used to distribute other keys
Speed	Fast	Slow, typically by a factor of up to 10,000 times slower than symmetric algorithms

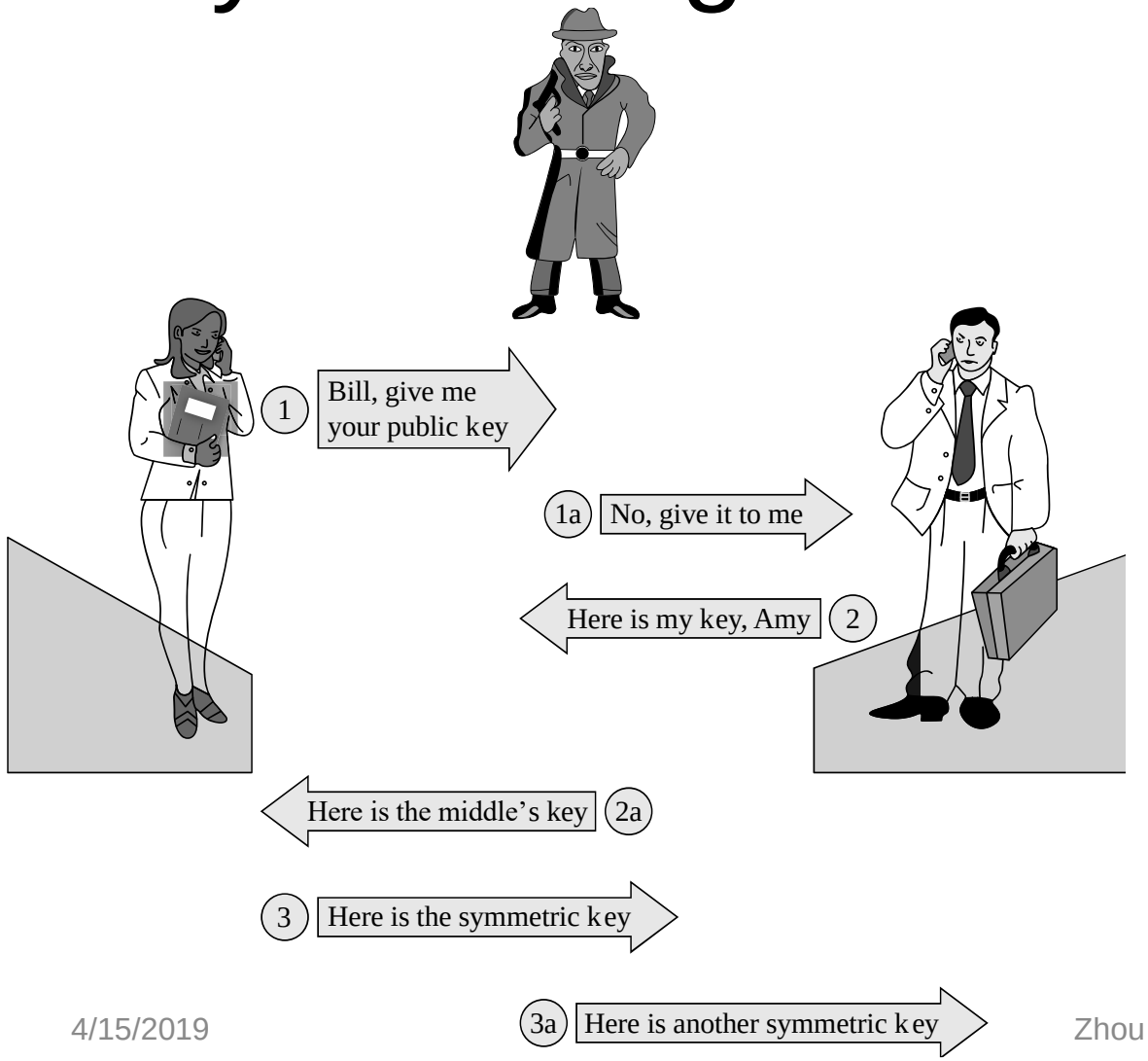
Public Key to Exchange Secret Keys



Question

- What's the problem of this paradigm?

Key Exchange Man in the Middle

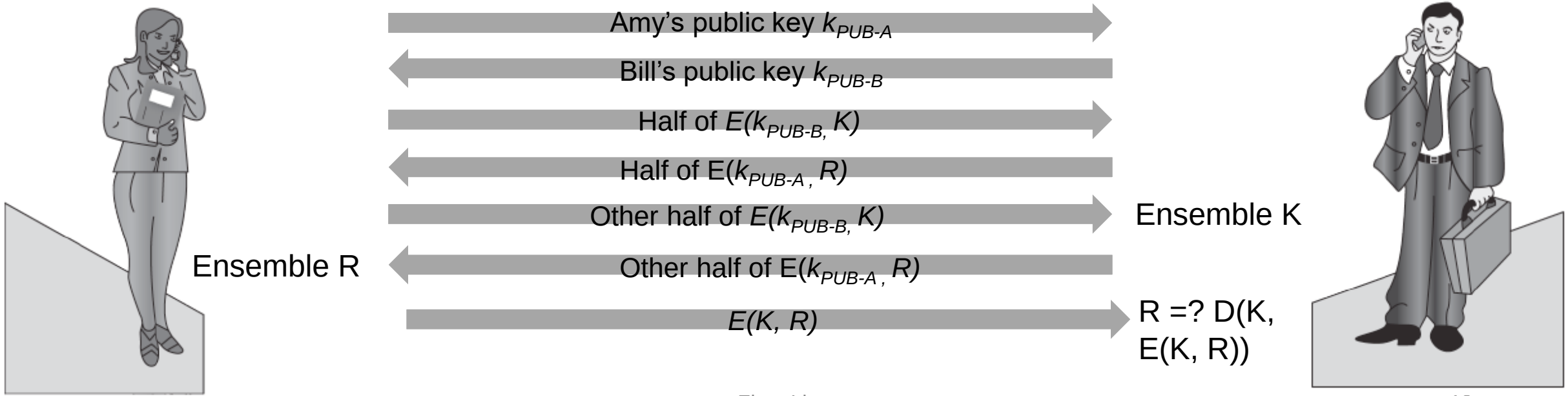


- Having both symmetric keys, Mallory can decrypt anything received, modify it, encrypt it under the other key, and transmit the modified version to the other party.
- Neither Amy nor Bill is aware of the switch.

Solution 1: Revised Key Exchange Protocol

- Rivest & Shamir [RIV84]
  - Amy and Bill exchange half a key at a time (first half, all odd, …).
  - Half a key is useless to the intruder because it is not enough to encrypt or decrypt anything.

K: symmetric key R: random number

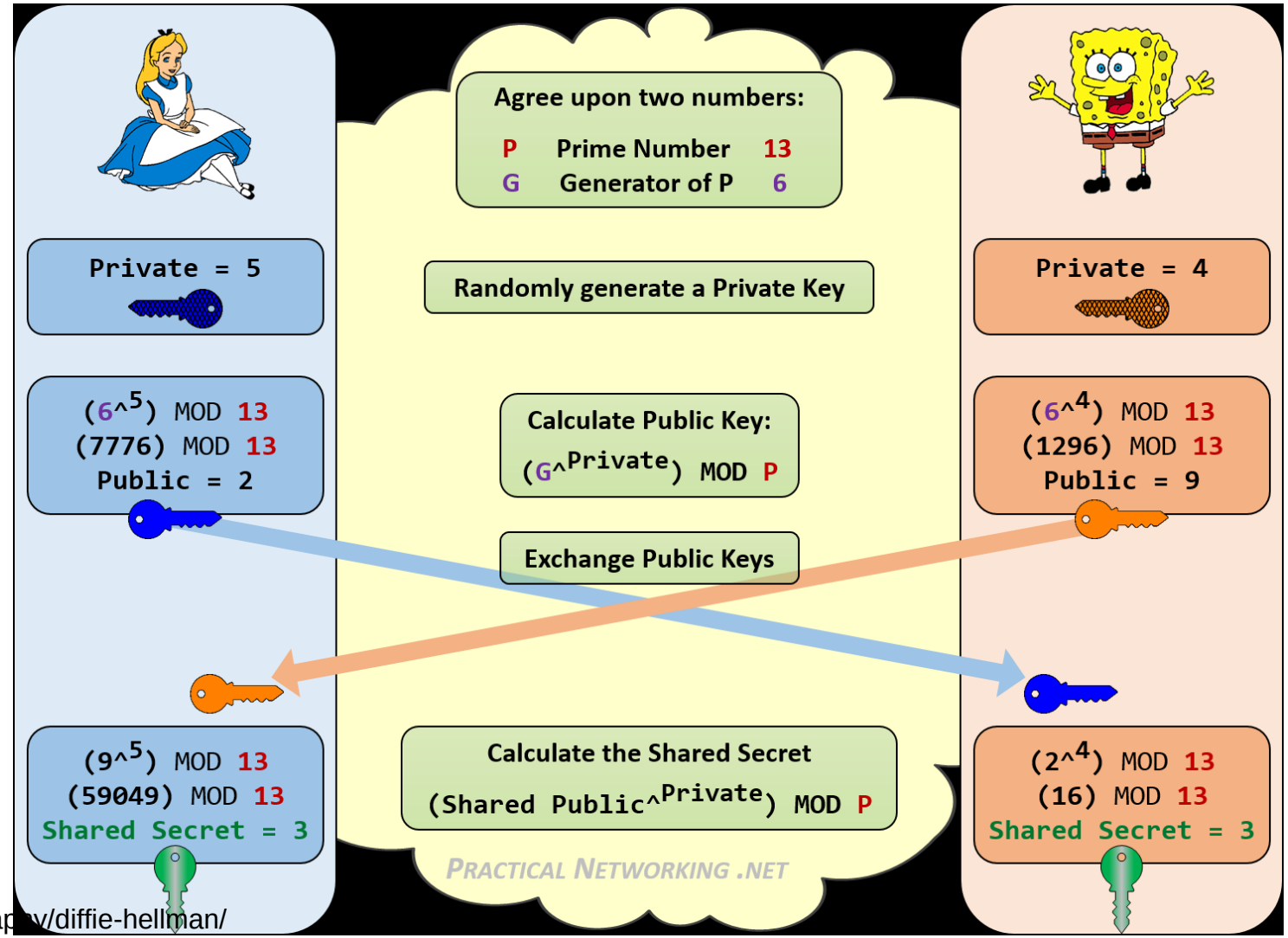


Solution 2: Authenticity

- Or, Amy should send to Bill $E(k_{PUB\text{-}B}, E(k_{PRIV\text{-}A}, K))$
  - Bill by using $k_{PRIV\text{-}B}$ can remove the encryption applied with $k_{PUB\text{-}B}$
  - Bill by using $k_{PUB\text{-}A}$ can remove the encryption applied with $k_{PRIV\text{-}A}$
  - Ensure only authentic party can see K
- Like lock (receiver) and seal (sender)

Solution 3: Diffie–Hellman key exchange

- Introduced at 1977
- One year before RSA



https://www.practicalnetworking.net/series/cryptography/diffie-hellman/

Solution 4: PKI

- PKI: public key infrastructure
  - Alice/Amy and Bill/Bob can verify the identity of sender
  - It will be covered later

Message integrity

- Prior primitives achieve confidentiality
- Need also to defend against active (tampering) attacks.
- Error Detecting Codes
  - Demonstrates that a block of data has been modified
- Simple error detecting codes:
  - Parity checks
  - Cyclic redundancy checks
- Cryptographic error detecting codes:
  - One-way hash functions
  - Cryptographic checksums
  - Digital signatures

Parity check

- An extra bit whose value depends on the sum of bits of original data
- Doesn't detect two-bit error
- Doesn't tell which bits are changes
- Useless when both original data and parity bit are modified

Original Data	Parity Bit	Modified Data	Modification Detected?
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 0 1	Yes
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 0	Yes
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 1	No
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 1 1	No
0 0 0 0 0 0 0 0	1	0 0 0 0 0 1 1 1	Yes
0 0 0 0 0 0 0 0	1	0 0 0 0 1 1 1 1	No
0 0 0 0 0 0 0 0	1	0 1 0 1 0 1 0 1	No
0 0 0 0 0 0 0 0	1	1 1 1 1 1 1 1 1	No